UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/688,573	10/20/2003	Robert M. Zeidman		2483

66323 7590 03/13/2007

ZEIDMAN TECHNOLOGIES, INC.
15565 SWISS CREEK LANE
CUPERTINO, CA 95014

EXAMINER	
WANG, BEN C	
ART UNIT	**PAPER NUMBER**
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	03/13/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/688,573	ZEIDMAN, ROBERT M.	
	Examiner	Art Unit	
	Ben C. Wang	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE ____ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 January 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-7 and 15-28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7, and 15-28* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated January 29, 2007, responding to the November 21, 2006 Office action provided in the rejection of claims 1-14, wherein claims 1-6 have been amended, claims 8-14 have been canceled, and new claims 15-28 have been added.

Claims 1-7, and 15-28 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection – see *Desmet et al.*, art made of record, as applied hereto.

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made

3. Claims 1-3, 5, 7, 15-17, 19, 21-24, 26, and 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lehman et al. (US Patent 4,796,179) (hereinafter 'Lehman') in view of Desmet et al. *'Operating System based Software Generation for Systems-on-Chip'*, 2000, ACM' (hereinafter 'Desmet' – art made of record)

4. **As to claim 1** (Currently Amended), Lehman discloses a method for developing a real-time operating system (Fig. 1; Col. 1, lines 46-48; Col. 4, lines 63-68; Col. 5, lines 1-2), comprising: specifying a set of *n* tasks (Col. 1, lines 33-38), *task(1)* through *task(n),* to be scheduled for execution; synthesizing source code for controlling execution of the n tasks (Abstract, lines 8-14; Col. 3, lines 1-8; Col. 5, lines 5-12; Col. 135, lines 17-24).

Lehman also discloses specifying a scheduling algorithm (Col.3, lines 36-39; Col. 9, lines 56-61; Col. 32, lines 44-47; Col. 20, line 63 through Col. 21, line 20; Col.7, lines 29-32; Col. 16, lines 21-23; Col. 2, lines 36-39; Col. 9, lines 62-68;

Col. 10, lines 1-2; Col. 32, lines 5-54) for scheduling the execution of the set of *n* tasks.

Although Lehman discloses at least one of the task of the set of *n* tasks being selected as a preemptive or a non-preemptive task (Col. 9, lines 52-56; Col. 10, Lines 3-12; Col. 35, Lines 1-5; Col. 136, Lines 40-50); synthesizing source code to implement a task scheduler (Fig. 4, element 24; Fig. 24; Fig. 26; Col. 2, lines 36-39; Col. 3, lines 36-39; Col. 9, lines 56-61; Abstract, lines 20-25; Col. 10, lines 8-12) that uses the scheduling algorithm for controlling execution of said *n* tasks.

But Lehman does not explicitly disclose the task of the set of *n* tasks being selectively configurable as a preemptive or a non-preemptive task; synthesizing source code with embedded commands to implement a task scheduler that uses the scheduling algorithm and the embedded commands for controlling execution of said *n* tasks.

However, in an analogous art, Desmet discloses the task of the set of *n* tasks being selectively configurable as a preemptive or a non-preemptive task (Sec. 2.4 – How Is It Implemented, 1st Para. – an internal object-oriented data structure is built at run-time, which stores the system configuration that is created using the API calls; this data structure can support analysis and synthesis (through code generation); synthesizing source code with embedded commands (Sec. 2.5.2, 1st Para. – SoCOS provides a system-call *soc_set_event_handler* that defines a reactive process; Fig. 1 – example of an asynchronous model (source code); Fig. 2 – example of a reactive mode (source code)) to implement

a task scheduler that uses the scheduling algorithm and the embedded commands (Fig. 1 – *Soc_sync_generator::run();* Fig. 2 - *Soc_react::run();* Sec. 2.5.1 – Asynchronous Models, 2nd Para.; Sec. 2.5.2 – Reactive Models, 2nd Para.; 5th Para.; Sec. 2.5.3 – Synchronous Models, 2nd Para.) for controlling execution of said *n* tasks.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Desmet into the Lehman's system to further provide the task of the set of *n* tasks being selectively configurable as a preemptive or a non-preemptive task; synthesizing source code with embedded commands to implement a task scheduler that uses the scheduling algorithm and the embedded commands for controlling execution of said *n* tasks in Lehman system.

The motivation is that it would enhance the Lehman's system by taking, advancing and/or incorporating Desmet's system which is the fully committed software model containing all information to generate the RTOS-based application source code based on the system configurable attributes, and using RTOS system API call; the C++ SoCOS API library for system level design offers the designer with services analogous to an operating system in software design; real-time aspects can be gradually introduced without rewriting the code by adding implementation configurable attributes and using RTOS system API calls, and further elaborating the refinement steps lead from the system level model to embedded real-time software source code as once suggested by Desmet (Sec.

2.4 – how is it implemented, 1st Para.; Sec. 5.3 – embedded software generation, 1st Para.; Sec. 5.4 - results, 2nd Para.; Sec. 6 - conclusion).

5. **As to claim 15** (news), Lehman discloses an apparatus for developing a real-time operating system comprising: a computer (Fig. 27 – multi-process controller, Lines 30-32); a computer readable medium in data communication with the computer (Col. 135, Line 15 through Col. 137, Lines 64), the computer readable medium including a software synthesis program stored thereon (Col. 135, Line 15 through Col. 137, Lines 64), which when executed by the computer causes the computer

Although Lehman discloses to specify a set of *n* tasks (Col. 1, lines 33-38), task (1) through task (n), to be scheduled for execution, at least one of the tasks of the set of *n* tasks (Abstract, lines 8-14; Col. 3, lines 1-8; Col. 5, lines 5-12; Col. 135, lines 17-24) as a preemptive or a non-preemptive task (Col. 9, lines 52-56; Col. 10, Lines 3-12; Col. 35, Lines 1-5; Col. 136, Lines 40-50); specify a scheduling algorithm (Col.3, lines 36-39; Col. 9, lines 56-61; Col. 32, lines 44-47; Col. 20, line 63 through Col. 21, line 20; Col.7, lines 29-32; Col. 16, lines 21-23; Col. 2, lines 36-39; Col. 9, lines 62-68; Col. 10, lines 1-2; Col. 32, lines 5-54) for scheduling the execution of the set of *n* tasks; and synthesize source code with to implement a task scheduler (Fig. 4, element 24; Fig. 24; Fig. 26; Col. 2, lines 36-39; Col. 3, lines 36-39; Col. 9, lines 56-61; Abstract, lines 20-25; Col. 10, lines 8-12) that uses the scheduling algorithm and for controlling execution of the set of *n* tasks.

But Lehman does not explicitly disclose specifying a set of *n* tasks, task (1) through task (n), to be scheduled for execution, at least one of the tasks of the set of *n* tasks being selectively configurable as a preemptive or a non-preemptive task; specify a scheduling algorithm for scheduling the execution of the set of *n* tasks; and synthesize source code with embedded commands to implement a task scheduler that uses the scheduling algorithm and the embedded commands for controlling execution of the set of *n* tasks.

However, in an analogous art, Desmet discloses specifying a set of *n* tasks, task (1) through task (n), to be scheduled for execution, at least one of the tasks of the set of *n* tasks being selectively configurable as a preemptive or a non-preemptive task (Sec. 2.4 – How Is It Implemented, 1st Para. – an internal object-oriented data structure is built at run-time, which stores the system configuration that is created using the API calls; this data structure can support analysis and synthesis (through code generation); specify a scheduling algorithm for scheduling the execution of the set of *n* tasks; and synthesize source code with embedded commands to implement a task scheduler that uses the scheduling algorithm (Sec. 2.5.2, 1st Para. – SoCOS provides a system-call *soc_set_event_handler* that defines a reactive process; Fig. 1 – example of an asynchronous model (source code); Fig. 2 – example of a reactive mode (source code)) to implement a task scheduler that uses the scheduling algorithm and the embedded commands (Fig. 1 – *Soc_sync_generator::run();* Fig. 2 - *Soc_react::run();* Sec. 2.5.1 – Asynchronous Models, 2nd Para.; Sec. 2.5.2 – Reactive Models, 2nd Para.; 5th Para.; Sec. 2.5.3 – Synchronous Models, 2nd

Para.) and the embedded commands for controlling execution of the set of *n* tasks (Fig. 1 – *Soc_sync_generator::run();* Fig. 2 - *Soc_react::run();* Sec. 2.5.1 – Asynchronous Models, 2nd Para.; Sec. 2.5.2 – Reactive Models, 2nd Para.; 5th Para.; Sec. 2.5.3 – Synchronous Models, 2nd Para.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Desmet into Lehman's system to further specify a set of *n* tasks, task (1) through task (n), to be scheduled for execution, at least one of the tasks of the set of *n* tasks being selectively configurable as a preemptive or a non-preemptive task; specify a scheduling algorithm for scheduling the execution of the set of *n* tasks; and synthesize source code with embedded commands to implement a task scheduler that uses the scheduling algorithm and the embedded commands for controlling execution of the set of *n* tasks in Lehman system.

The motivation is that it would enhance the Lehman's system by taking, advancing and/or incorporating Desmet's system which is the fully committed software model containing all information to generate the RTOS-based application source code based on the system configurable attributes, and using RTOS system API call; the C++ SoCOS API library for system level design offers the designer with services analogous to an operating system in software design; real-time aspects can be gradually introduced without rewriting the code by adding implementation configurable attributes and using RTOS system API calls, and further elaborating the refinement steps lead from the system level model to embedded real-time software source code as once suggested by Desmet (Sec.

2.4 – how is it implemented, 1st Para.; Sec. 5.3 – embedded software generation, 1st Para.; Sec. 5.4 - results, 2nd Para.; Sec. 6 - conclusion).

6. **As to claim 22** (news), Lehman discloses an apparatus for developing a real-time operating system (Fig. 1; Col. 1, lines 46-48; Col. 4, lines 63-68; Col. 5, lines 1-2) comprising: b) means for specifying a scheduling algorithm for scheduling the execution of said the of *n* tasks (Col.3, lines 36-39; Col. 9, lines 56-61; Col. 32, lines 44-47; Col. 20, line 63 through Col. 21, line 20; Col.7, lines 29-32; Col. 16, lines 21-23; Col. 2, lines 36-39; Col. 9, lines 62-68; Col. 10, lines 1-2; Col. 32, lines 5-54).

Although Lehman discloses a) means for specifying a set of n tasks, task (1) through task (n), to be scheduled for execution, at least one of the tasks of the set of n tasks being a preemptive or a non-preemptive task; c) means for synthesizing source code with to implement a task scheduler that uses said scheduling algorithm and said for controlling execution of the set of *n* tasks.

But Lehman does not explicitly disclose a) means for specifying a set of n tasks, task (1) through task (n), to be scheduled for execution, at least one of the tasks of the set of n tasks being selectively configurable as a preemptive or a non-preemptive task; c) means for synthesizing source code with embedded commands to implement a task scheduler that uses said scheduling algorithm and said embedded commands for controlling execution of the set of *n* tasks.

However, in an analogous art, Desmet discloses a) means for specifying a set of n tasks, task (1) through task (n), to be scheduled for execution, at least

one of the tasks of the set of n tasks being selectively configurable as a preemptive or a non-preemptive task (Sec. 2.4 – How Is It Implemented, 1st Para. – an internal object-oriented data structure is built at run-time, which stores the system configuration that is created using the API calls; this data structure can support analysis and synthesis (through code generation); c) means for synthesizing source code with embedded commands to implement a task scheduler that uses said scheduling algorithm (Sec. 2.5.2, 1st Para. – SoCOS provides a system-call *soc_set_event_handler* that defines a reactive process; Fig. 1 – example of an asynchronous model (source code); Fig. 2 – example of a reactive mode (source code)) and said embedded commands for controlling execution of the set of *n* tasks (Fig. 1 – *Soc_sync_generator::run();* Fig. 2 - *Soc_react::run();* Sec. 2.5.1 – Asynchronous Models, 2nd Para.; Sec. 2.5.2 – Reactive Models, 2nd Para.; 5th Para.; Sec. 2.5.3 – Synchronous Models, 2nd Para.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Desmet into Lehman's system to further a) means for specifying a set of n tasks, task (1) through task (n), to be scheduled for execution, at least one of the tasks of the set of n tasks being selectively configurable as a preemptive or a non-preemptive task; c) means for synthesizing source code with embedded commands to implement a task scheduler that uses said scheduling algorithm and said embedded commands for controlling execution of the set of *n* tasks in Lehman system.

The motivation is that it would enhance the Lehman's system by taking, advancing and/or incorporating Desmet's system which is the fully committed software model containing all information to generate the RTOS-based application source code based on the system configurable attributes, and using RTOS system API call; the C++ SoCOS API library for system level design offers the designer with services analogous to an operating system in software design; real-time aspects can be gradually introduced without rewriting the code by adding implementation configurable attributes and using RTOS system API calls, and further elaborating the refinement steps lead from the system level model to embedded real-time software source code as once suggested by Desmet (Sec. 2.4 – how is it implemented, 1st Para.; Sec. 5.3 – embedded software generation, 1st Para.; Sec. 5.4 - results, 2nd Para.; Sec. 6 - conclusion).

7. **As to claims 2** (Currently Amended), **and 23** (New), Lehman discloses the method further including specifying *t init-tasks* that are executed only once upon initial execution of said task scheduler, *t* being less than or equal to *n* (Col. 3, lines 36-39; Col. 9, lines 56-61 – when the execution of each code segment is initialized; Col. 32, lines 44-47).

8. **As to claims 3** (Currently Amended), **and 24** (New), Lehman discloses the method and the apparatus further including specifying *f f-loop* tasks, each having an associated integer value *li* for *i* ranging from *1* to *f* and *f* being less than or equal to *n* (Col. 20, line 63 through Col. 21, line 20 – for loops using an

incrementing or decrementing counter, i.e. Loop for *I* = 1 to *X* (executing) block of statements), said task scheduler addresses the task scheduler executing the loops including a continuously executing loop such that each *f-loop* task executes exactly once every *li* times that the loop is executed (Col. 21, lines 13-19).

9. **As to claims 5** (Currently Amended)**, 19** (New), **and 26** (New), Lehman discloses the method and apparatus further including specifying *c call-tasks*, *c* being less than or equal to *n*, the task scheduler scheduling a *call-task* when another task requests that the *call-task* be executed (Col. 7, lines 29-32; Col. 16, lines 21-23).

10. **As to claims 7** (Previously Presented) **21** (New), **and 28** (New), Lehman discloses the method and the apparatus where tasks are given priority values such that whenever the task scheduler chooses between scheduling multiple tasks, all of which being ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values (Col. 2, lines 36-39; Col. 9, lines 62-68; Col. 10, lines 1-2; Col. 32, lines 5-54).

11. **As to claim 16** (New), Lehman discloses the apparatus being further configured to specify *t* init-tasks that are executed only once upon initial execution of said task scheduler, *t* being less than or equal to *n* (Col. 3, lines 36-39; Col. 9, lines 56-61 – when the execution of each code segment is initialized; Col. 32, lines 44-47).

12. **As to claim 17** (New), Lehman discloses the apparatus being further configured to specify *f f-loop* tasks, each having an associated integer value *I(i)* for *i* ranging from *1* to *f* and *f* being less than or equal to *n* (Col. 20, line 63 through Col. 21, line 20 – for loops using an incrementing or decrementing counter, i.e. Loop for *I* = 1 to *X* (executing) block of statements), the task scheduler including a continuously executing loop such that each *f-loop* task executes exactly once every *I(i)* times that the loop is executed (Col. 21, lines 13-19).

13. Claims 4, 18, and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lehman, in view of Desmet and in further view of Xu et al. *On Satisfying Timing Constraints in Hard-Real-Time Systems'*, 1991, ACM' (hereinafter 'Xu')

14. **As to claims 4** (Currently Amended), **18** (New Added), **and 25** (New), Lehman discloses the method and apparatus including means for specifying "loops" mechanism (Col. 20, line 63 through Col. 21 line 20).

But, Lehman and Desmet do not specifically disclose *p-loop* task. However, in an analogous art, Xu discloses means for specifying *p-loop* tasks, each having an associated integer value *ti* for *i* ranging from *1* to *p* and *p* being less than or equal to *n*, the number *ti* representing a number of regular time units (Sec. 2, 3rd paragraph, lines 1-4), said task scheduler including a timer that schedules each *p-loop* task i to be executed approximately once every *ti* time

units (Sec. 2, 3rd paragraph, lines 1-4; Sec. 2, 7th paragraph, on page 133 – A periodic process *p* can be described by a quadruple(*rp, cp, dp, prdp*), where *prdp* is the period, *cp* is the worse case computation time required by process *p*, *dp* is the deadline, *rp* is the release time).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Lehman and Desmet, and the teachings of Xu in order to provide a timing constraints mechanism in Lehman-Desmet system.

The motivation is that (a) pre-run-time scheduling is essential if we want to guarantee that timing constraints will be satisfied in a complex hard-real-time system, (b) appropriate algorithms for solving mathematical scheduling problems that address those concerns can be used to automate pre-run-time scheduling, (c) if the task of computing schedules is completely automated, it would be very easy to modify the system and re-compute new schedules in case changes are required by applications as once suggested by Xu (Abstract, Lines 1-3; Sec. 6, 3rd Para., 6th Para.).

15. Claims 6, 20, and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lehman, Desmet in view of Xu and further in view of D. Lake (US 2004/0045003 A1) (hereinafter 'Lake'),

16. **As to claims 6** (Currently Amended), **20** (New), **and 27** (New), Lehman discloses the method and the apparatus including means for further specifying *r*

preemptive- tasks (Col. 9, lines 52-56), *r* being less than or equal to *n*, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing (Col. 35, lines 7-14) and continuing the execution of preemptive-task (Col. 9, line 64 through Col. 10, line 2).

But Lehman, Desmet, and Xu do not specifically disclose the task's state is stored and execution is given to the task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task. However, in an analogous art, Lake discloses the task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task

However, in an analogous art, Lake discloses the task's state is stored and execution is given to the task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task. However, in an analogous art, Lake discloses the task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task (Fig. 1; [0031]; [0026], lines 1-9; [0036], lines 1-6).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Lehman, Desmet, and Xu with the teachings of Lake in order to save/restore task control data during a preemptive-task interruption and task resumed in Lehman-Desmet-Xu system.

The motivation is to have its stack pointer set to a pre-calculated worst-case value guaranteed to leave sufficient space in the stack beneath the stack pointer for any preemptive tasks for task suspended/restored operations as once suggested by Lake (i.e., Abstract).

Conclusion

17. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- C. L. Liu, Scheduling Algorithms for multiprogramming in a Hard-Real-Time Environment, January 1, 1973, ACM
- Desmet et al., Operating System based Software Generation for System-on-Chip, 2000, ACM
- Desmet et al., Timed Executable System Specification of an ADSL Modem using a C++ based Design Environment: A Case Study, 1999, ACM
- Verkest et al., On the use of C++ for system-on-chip design, 1999, IEEE

18. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax

phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TUAN DAM
SUPERVISORY PATENT EXAMINER

BCW

February 28, 2007